Parallel Algorithms for the Simulation of Protobiological Membranes

Joint Research Interchange no. NCA2-792

Final Technical Report

Michael A. Wilson and Andrew Pohorille
Department of Pharmaceutical Chemistry
University of California, San Francisco
San Francisco, CA 94143

and

Sherwood Chang
NASA – Ames Research Center
Moffett Field, CA 94035

Introduction

In the origin of life on Earth, the development of the boundary between the cellular interior and the external environment was crucial for much of the structure and functions that exist in modern cells. The first structures which had a cell–like structure were probably vesicles. These are closed, spheroidal aggregates of amphiphilic molecules arranged in a bilayer. Amphiphilic molecules are composed of polar or charged headgroups and long oily tails. In bilayers, the amphiphilic molecules are arranged with the headgroups pointing toward the aqueous medium and the oily tails form the interior of the bilayer. Thus, the bilayer provides an "oil–like" barrier between the aqueous interior and the aqueous exterior.

The principle role of vesicles in the origin of life was to protect molecules located in the aqueous interior from degradation and dilution in the external environment. However, vesicles alone cannot be considered to be protocells because protocellular function requires additional features such as (1) the capture and transduction of energy, (2) the transport of material across the membrane boundary, and (3) the ability to catalyze the synthesis of new protocellular material. In the absence of the complicated protein molecules that regulate modern cellular activities, these functions must have been performed by much simpler mechanisms in protocells. One fundamental feature that protocells must have shared with modern cells is that cellular processes were closely coupled in both time and space. As in contemporary cells, these processed most likely occurred in the membrane interior or at the membrane surface.

Molecular level computer simulations provide a powerful and natural tool for investigating protocellular phenomena. These methods yield a detailed and accurate microscopic description of a molecular system that is often not available experimentally. In the context of the origin of life, we have applied molecular dynamics computer simulations of amphiphilic molecules and ions at aqueous interfaces and, more recently, of water–oil interfaces and planar water–bilayer interfaces. These studies yielded much new structural

information. Molecular dynamics has been also broadly used to investigate components of contemporary cells, such as proteins and nucleic acids and have proved to be helpful in fields of molecular biology and drug design. However, currently available computer resources have limited applications of simulation methods to studies of single molecules, interactions between pairs of molecules or, at best, small molecular aggregates.

Our present research entails developing parallel algorithms for computational studies of bilayer membrane systems. These algorithms must efficiently treat very large, anisotropic, multicomponent systems. These systems typically consist of several types of particles (atoms, molecules) unevenly distributed in space and interacting via different forces. Algorithms for homogeneous, isotropic systems, which have recently been proposed, are not expected to be appropriate in this case, because the spatial inhomogeneities give rise to load–balancing problems in the computations. Here, we discuss parallel algorithms that provide better load–balancing and test these algorithms in simulations of simple aqueous interfaces.

Methods

A Molecular Dynamics (MD) simulation requires the solution of Newton's equations of motion for each atom in the system. Solving Newton's equations of motion requires knowledge of the forces acting on all the atoms in the system. These forces are obtained as the derivatives of the potential energy functions describing the interactions between all pairs of atoms. This is usually the most time–consuming step in MD simulations. A system with N atoms has $N(N-1)/2$ interatomic interactions that must be evaluated at every time step. In practical applications this number is reduced by applying a spherical cutoff, r_{cut}, whereby interactions between atoms separated by more than r_{cut} are neglected. This reduces the time required for force evaluation at each step from $O(N^2)$ to $O(Nr_{cut}^2)$. The main consideration in choosing r_{cut} is to ensure that the omitted interactions are small and unimportant to the process studied.

In a typical MD simulation, about 95% of the CPU time is devoted to the calculation of the forces. Efficient use can be made of both vector and parallel architectures in the force calculation, and MD simulations of 10^6 single-component atoms have been reported in the literature. In a system which is homogeneous and isotropic, and in which the force cutoff r_{cut} is short relative to the size of the simulation box, a parallel link–cells (PLC) algorithm has been shown to provide an efficient method for the force calculation. In the PLC algorithm the simulation box is divided into P identical cells, where P is the number of processors. All the molecules in a particular cell are assigned to the corresponding processor. Each cell is then divided into a total of M subcells which are of about the same size as r_{cut}. In general, the total number of subcells is much larger than the number of processors, P. The basic feature of the PLC algorithm is that in the hypercube architecture it requires communication only between cells (processors) that are directly adjacent. This can be performed in a highly coordinated manner.

Results

In the current study, the system was restricted to the water liquid–vapor interface containing 500 water molecules. This is a single component, spatially inhomogeneous system. The code was ported to the CM–5 using High Performance Fortran (HPF) constructs (specifically, the CM Fortran extensions of Fortran 90 and HPF). This method offers some chance of portability, although this is limited at the present time by the number of available HPF compilers. Using this method, the distribution of the particles across the processors is handled using $LAYOUT directives in the HPF Code.

In a typical force calculation, the computation contains an outer loop, i, over the atoms (or molecules) from 1 to $N-1$, and an inner loop from $i+1$ to N. For large values of i, the inner loop can become quite small, making it inefficient on vector or parallel machines. One possible solution is to "stack" molecules into temporary arrays, and do the force calculation over these stacks. Not only are the stacks longer, which aids in vector

performance, but they are more uniform in lenth, which aids in parallel performance. Table I shows the amount of time spent in the force loop calculation on 32 Nodes of a CM-5 for several different stack sizes.

Table I. Force loop computation for various stack sizes on a CM–5.

size	time (sec/step)
500	0.273
1000	0.148
2000	0.085
10000	0.042
20000	0.036

While it might appear that the very large stack sizes are most efficient, there are additional costs that have not been addressed. First, there is some overhead associated with the construction of each stack. Longer stacks, in general, require greater amounts of communications between the nodes. In addition, once the forces have been calculated in a given stack, these must be distributed back to the actual force arrays (which have the same distribution across the nodes as the molecule arrays). Here, too, the larger stacks lead to greater time spent in interprocessor communication. At least for the present calculation, it appears that stack sizes on the order if 1000 to 2000 molecules appears optimal. It is also possible that the optimal distribution of the molecules across the depends upon the stack size, and that better ways can be found to construct the assignment of the molecules to the processors.

Overall, the force computation compares favorably with other machines. Unfortunately, we only have timings for the entire force (including the overhead associated with

the setup and distribution of the forces) required to calculation on th other machines, so a direct comparison can't be made. The time required for the force calculation on different machines is shown in Table II:

Table II. Total force calculation for various platforms.

Machine	time (sec/step)
SGI 4D/240	2.8
DEC 3000/800	0.25
Cray C90	0.097
CM-5 (32 nodes)	0.085[a]

[a]This number does not include gather/scatter overhead.

Conclusions

Computer simulations of a water liquid–vapor interfacial system have been carried out on a Connection Machine CM–5. These calculations served primarily as a test of parallel algorithms for the calculation of forces in spatially inhomogeneous systems. While the HPF constructs are perhaps the simplest way to port code from a scalar to a parallel machine, the actual assignment of particles to different processors is somewhat difficult. As a consequence, more time is spent in communication than necessary in an explicit message passing program. A message passing version of the code is under development, and we expect to compare these results with the present HPF results during the extension of the JRI.

In a bilayer vesicle, or many other systems of interest, the system is neither isotropic nor homogeneous. A general extension of the PLC algorithm to multi-component, anisotropic systems which maintains a proper load-balancing among the processors is required. This algorithm is also being developed and tested at the present time. The PLC algorithm will then be used in conjunction with fast, multipole expansion methods for calculating long-ranged forces. Together, these should provide an efficient method for molecular dynamics calculations of bilayer systems on parallel architectures.

Acknowledgements

This work has been supported in part by a NASA - Ames/UC San Francisco Joint Research Initiative, NCA2-792, and the hospitality of the Department of Chemistry, U. C. San Francisco and the Planetary Biology Branch of the NASA - Ames Research Center. Computer resources for this work were provided in part by the Numerical Aerodynamical Simulation (NAS) program.